Exhibit 1

Method and System for Machine-Aided Rule Construction for Event Management

U.S. PATENT DOCUMENTS

•5,874,955	3/12/96	Rogowitz, Rabenhorst, Treinish
•5,661,668	8/26/97	Yemini; Yechiam ,Yemini; Shaula, Kliger; Shmuel ("Apparatus and method for analyzing and correlating events in a system using a causality matrix")

OTHER PUBLICATIONS

- Computer Associates International, "Neugents. The Software that can Think," July 16, 1999, http://www.cai.com/neugents.
- Sheng Ma and Joseph L. Hellerstein, ``EventBrowser: A Flexible Tool for Scalable Analysis of Event Data," Distributed Operations and Management, 1999.
- K.R. Milliken et al. "YES/MVS and the automation of operations for large computer complexes," IBM Systems Journal, Vol 25, No. 2, 1986.
- Tom M. Mitchell. **Machine Learning**. McGraw Hill, 1997.

ABSTRACT

Methods and systems are described for learning correlation rules used in event management. The key method consists of the steps of (a) marking one or more event groupings; (b) employing a machine learning program to learn the underlying concept of these groupings; (c) including a rule right-hand side; and (d) putting the new rule in the Rule DB. The system to implement this method consists of components for: (1) interactive visualization and user interface control; (2) query-based learning; (3) Event DB access; and (4) correlation Rule DB access.

FIELD OF THE INVENTION

The present invention relates generally to network and systems management and more specifically to detecting and resolving availability and performance problems.

BACKGROUND

With the dramatic decline in the price of hardware and software, the cost of ownership for computing devices is increasingly dominated by network and systems management. Included here are tasks such as establishing configurations, help desk support, distributing software, and ensuring the availability and performance of vital services. The latter is particularly important since inaccessible and/or slow services decrease revenues and degrade productivity.

The first step in managing availability and performance is event management. Almost all computing devices have a capability whereby the onset of an exceptional condition results in the generation of a message so that potential problems are detected before they lead to widespread service degradation. Such exceptional conditions are referred to as **events**. Examples of events include: unreachable destinations, excessive CPU consumption, and duplicate IP addresses. An event message contains multiple attributes, especially: (a) the source of the event, (b) type of event, and (c) the time at which the event was generated.

Event messages are sent to an **event management system (EMS)**. An EMS has an **adaptor** that parses the event message and translate it into a normalized form (an adaptor). This normalized information is then placed into an **Event DB**. Next, the normalized event is fed into a **correlation engine** that that determines actions to be taken. This determination is typically driven by correlation rules that are kept in a **Rule DB**. Examples of processing done by correlation rules includes:

1. Elimination of duplicate messages. Duplicate is interpreted broadly here. For example, if multiple hosts on the same local area network generate a destination-unreachable message for the same destination, then the events contain the same information.

2. Maintenance of operational state. State may be as simple as which devices are up and which are down. It may be more complex as well, especially for devices that have many intermediate states or special kinds of error conditions (e.g., printers)

3. Problem detection. A problem is present if one or more components of the system are not functioning properly. For example, the controller in a load balancing system may fail in a way so that new requests are always routed to the same back-end web server, a situation that can be tolerated at low loads but can lead to service degradation at high load. Providing early detection of such situations is important in order to ensure that problems do not lead to wide-spread service disruptions.

4. Problem isolation. This involves determining the components that are causing the problem. For example, distributing a new release of an application that has software errors can result in problems for all end-users connecting to servers with the updated application. Other examples of problem causes include: device failure, exceeding some internal limit (e.g., buffer capacity), and excessive resource demands.

The correlation engine provides automation that is essential for delivering cost effective management of complex computing environments. Current art provides three kinds of correlation. The first employs operational policies expressed as rules (e.g., Milliken, 1986). Rules are if-then statements in which the if-part tests the values of attributes of individual event, and the then-part specifies actions to take. An example of such a rule is "If a hub generates an excessive number of interface-down events, then check if the software loaded on the hub is compatible with its hardware release." The industry experience has been that such rules are difficult to construct, especially if they include installation-specific information.

Another approach has been developed by SMARTS (Yemini et al.) based on the concept of a codebook that matches a repertoire of known problems with event sequences observed during operation. Here, operational policies are models of problems and symptoms. Thus, accommodating new problems requires properly modeling their symptoms and incorporating their signatures into the code book. In theory this approach can accommodate installation-specific problems. However, doing so in practice is difficult because of the high level of sophistication required to encode installation-specific knowledge into rules.

Recently, a third approach to event correlation has been proposed (Computer Associates International, 1999). This approach trains a neural network to predict future occurrences of events based on the frequency of their occurrence in historical data. Typically, events are specified based on thresholds, such as CPU utilization exceeding 90%. The policy execution system uses the neural network to determine the likelihood of one of the previously specified events occurring at some time in the future. While this technique can provide advanced knowledge of the occurrence of an event, it still requires

specifying the events themselves. At a minimum, such a specification requires detailing the following:

1. The variable measured (e.g., CPU utilization)

2. The directional change that considered (e.g., too large)

3. The threshold value (e.g., 90%)

The last item can be obtained automatically from examining representative historical data. Further, graphical user interfaces can provide a means to input the information in items (2) and (3). However, it is often very difficult for installations to choose which variables should be measured and the directional change that constitutes an exceptional situation.

To summarize, current art for event manage systems is of three types. The first (e.g., as in Milliken, 1986) requires that correlation rules be specified by experts, a process that is time-consuming and expensive. The second (e.g., as in Yemini) reduces the involvement of experts but only for aspects of event management that share broad commonalties (e.g., IP connectivity). The third (e.g., Computer Associates International, 1999) attempts to automate the construction of correlation rules for a broader range of management areas. However, to date, this has not been done in a manner that provides for customization by experts, especially in a way that avoids dealing with low-level details (e.g., specific threshold values, the choice of measurement values, and directional changes of interest for these variables).

More broadly, no existing EMS provides decision support for constructing correlation rules, which we refer to as **machine-aided rule construction**. At best, existing art provides authoring systems that aid in syntax checking and formatting. However, no assistance is provided for translating examples of event patterns (drawn from historical data) into correlation rules.

SUMMARY OF THE INVENTION

The present invention addresses the problem of decision support for constructing correlation rules for event management. The invention includes

systems and methods for visualizing event data and machine-based processing of these data to aid in rule construction.

Providing decision support for rule construction requires capabilities for visualizing and describing patterns in terms of rule left-hand sides. In the area of visualization, our starting point is the work described in Ma and Hellerstein, 1999. Also relevant here is Rogowitz, Rabenhorst, and Treinish, 1999 which describes a way to provide preferred visualizations.

Our invention makes use of machine learning algorithms to describe patterns in terms of rules. The framework we adopt is learning concepts expressed as predicates on attributes (e.g., as in Mitchell, 1997). In essence, a concept is a where-clause as expressed in the structured query language (SQL). An example is

> *All events originate from subnet 15.2.3 and the* event rate exceeds .75 per second.

Here, the attribute subnet must have the value 15.2.3 and the total number of events divided by the time-span in seconds of the group must exceed .75.

Learning concepts is greatly facilitated by using one or more **abstraction hierarchy**. An example of a two level hierarchy is: (a) the host itself (e.g., yahoo.com) and (b) its type (e.g., file server, web server, name server). A more extensive hierarchy might be based on the kind of interactions (workload) being done such as: (i) the specific transaction, (ii) the user performing the transaction, (iii) the user's department, and (iv) the division in which the user works. If both cases, we have containment in that higher levels encompass lower ones. In event management, there are often multiple hierarchies (e.g., time, configuration, workload, event type).

A broad class of learning algorithms we consider (e.g., generalization-specialization algorithms as in Mitchell, 1997) uses abstraction hierarchies in two ways. First, when a positive example is encountered that is not covered by the current set of predicates, the level of one or more abstraction hiearchies is increased to include this example. Second, when a negative example is encountered that is covered by the predicate, the level of one or more abstraction hierarchies is decreased. Various schemes are used to optimize that hierarchy levels chosen to maximize the number of positive examples covered and minimize the number of negative examples covered..

The main method of our invention consists of a series of interactions between an analyst--a domain expert--and our system (hereafter, referred to as the machine) whereby correlation rules are constructed. This method consists of steps for (1) the analyst marking one or more event groupings; (2) the machine learning the left-hand side for event patterns; (3) the analyst adding the right-hand side; and (4) the machine placing the rule in the Rule DB that is used by the correlation engine of the EMS. Step (1) is greatly aided by having an effective tool for visualizing and interacting with event groups. Step (2) employs machine-learning techniques, especially query-based learning and generalization-specialization hierarchies that allow a machine to choose the best level of an abstraction hierarchy to cover positive examples of an event pattern (and avoid negative examples).

To elaborate, in our invention learning a left-hand side means determining the predicates necessary to describe a set of event groupings. Predicates consist of logical statements about attribute values. For example, it may be that event groups are characterized originating from hubs, on subnet 9.2.16, with an event rate of .5/second. Then, we want a learning algorithm to determine these predicates.

The foregoing method may be augmented by incorporating data mining algorithms to aid in finding patterns of interest. These patterns can, in essence, seed the process of finding left-hand sides of rules.

The system of our invention includes components for interactive visualization, learning event patterns, rule construction, event data access, and Rule DB access. The visualization system in conjunction with event data access provide a means for analysts to select event groupings that are then translated into left-hand sides by the pattern learner. Rule construction in combination with Rule DB access provide a means for adding the rule right-hand side and placing the result in the Rule DB.

Considerable benefits accrue from our invention. First, the construction of correlation rules is made easier in that left-hand sides of rules can be generated automatically. Clearly, this is a productivity benefit in that expressing left-hand sides can be time consuming. In addition, this capability can allow those who are knowledgeable about operations to develop rules even though they may not be trained in constructing correlation rules.

Another benefit of the present invention relates to the assessment of correlation rules once they are constructed. In existing art, rules are evaluated by using them in production systems. While in some sense this is the ultimate test, it may be some time before a situation arise when the rule is invoked. A

complementary approach is to apply the rule's left-hand side to historical data, selecting instances of the patterns specified by the rule. By so doing, the operations staff can determine if the situations for which the rule is intended are in fact those that will be selected in production.

BRIEF DESCRIPTION OF DRAWINGS

Fig. 1 shows the overall architecture in which our invention operates.

Fig. 2 displays a visualization used to identify groupings of events when learning the left-hand side of rules.

Fig. 3 shows the process for machine-aided rule construction.

Fig. 4 describes the process for query-based learning of a rule left-hand side.

Fig. 5 displays some hierarchies used in employing the generalization-specialization algorithm to learn rule left-hand sides.

Fig. 6 illustrates the system for machine-aided rule construction.

Fig. 7 shows the system in our invention for pattern learning

Fig. 8 displays a user interface for inputting the information in box 500 in Fig. 4.

Fig. 9 displays a user interface for presenting the results of box 520 in Fig. 4.

Fig. 10 displays a user interface for accomplishing box 530 of Fig. 4.

EMBODIMENT

Fig. 1 displays the overall architecture of the environment in which our invention operates. An operator (100) receives alerts and initiates responding actions based on interactions with the event management system (110) that receives events (170) from various devices, such as file servers (140), name servers (150), and mail servers (160). The event management system updates

the Event DB (180) with newly received events and reads this DB to do event correlation based on the Rule DB (185). An analyst (120) uses the event management decision support system (130) of our invention to develop the correlation rules used by the event management system to control the interactions with the operator. Doing so requires reading historical event data in the Event DB and writing the Rule DB.

Fig. 2 illustrates the kind of display used by the event management decision support system to identify patterns that should be translated into the left-hand side of rules. Many patterns may be present, including periodicities and event bursts. Also, patterns may exist at multiple time-scales.

Fig. 3 displays the process for machine-aided rule construction in our invention. Boxes beginning with an "A" are performed by an analyst (human); those that begin with an "M" are done by the machine; and those with "A,M" are done collaboratively by the analyst and the machine. In 410, the analyst and the machine collaborate to learn the left-hand side of rules based on patterns identified in visualizations such as those displayed in Fig. 2. In 420, the analyst augments the left-hand side with a right-side action. In 430, the machine places the new rule in the Rule DB of the event management system (which is box 330 in Fig. 6).

Fig. 4 provides the details of step 410 in Fig. 3. In 500, the analyst marks one or more event groupings and indicates if they are positive or negative examples of the problem to be detected in the rule's left-hand side. In 505, the machine learns a concept using the positive and negative examples provided by the analyst. In 510 the machine determines if there are a sufficient number of examples to learn the rule left-hand side. If there are, the flow proceeds to step 420. If there is not, the machine looks for similar patterns based on the rule constructed so far. In 530, the analyst critiques the result by determining if the examples to date accurately reflect the concept to be identified. This may involve: (a) reclassifying a positive example as a negative example or a negative example as a positive example; (b) deleting examples; and (c) including or excluding events in an example so that it better conforms with the concept being learned. In 540, the analyst may optionally adjust the parameters of the learner to better operate with the concept being learned. Then, the flow continues with box 505.

To elaborate on step 520, consider the preliminary concept "there is a port-down event followed by a port-up event from the same host in within 5 seconds". The machine seeks other examples of such an event sequence from a single host. One way this can be done is for the machine to do a SQL query that retrieves all event interface-down events. Then for each, the machine also retrieves the events that occurred over the next five seconds from that same

host. The machine then checks if one of these events is an interface-up. For those hosts that this is the case, the machine then reports the entire sequence of events from interface-down through interface-up.

Fig. 5 displays examples of the hierarchies used to learn concepts for the left-hand side of rules. Four hierarchies are shown. In 600, there is a time hierarchy consisting of work shift (610), hour of the day (625), and minute within the hour (630). In 605, there is a configuration element hierarchy consisting of the type of host (635) (e.g., mail server, file server) and the host identifier (640) (e.g., its Internet address). In 610, there is a workload hierarchy consisting of the division in which the user is employed (645), the department (650), and user's name (655), and the transaction (or work unit) being performed by the user (660). In 615, there is a hierarchy of event types consisting of the situation in which the event occurs (e.g., printer failure) (665), the nature of the action (670), and the specifics of the element itself (675).

Fig. 6 displays the components of the event management decision support system (130). The interactive visualization and user interface control (700) provides overall control of the interactions with the analyst (120) and the flow within the event management decision support system. The pattern learner (710) is invoked to perform step 410 described in Fig. 3, and the rule constructor (730) in collaboration with the Rule DB access component (725) are used to perform step 430 in Fig. 3, which involves reading and writing the Rule DB. The event miner (705) and event data access (720) components are used in combination to aid in visualization and similarity query functionality used in step 520 of Fig. 4, which requires reading the Event DB.

Fig. 7 details the elements of the pattern learner (710) in Fig. 6. The event visualization and control component (800) controls interactions with the analyst for purposes of learning event patterns. 800 also controls the flow within the pattern learner, including queries to the Event DB via the constraint query engine (805), which in turn invokes the Event Data Access component (810) to read from the Event DB. In addition, 800 invokes the Pattern Inference component (815) to determine possible patterns in the set of positive and negative examples, and establishes hierarchies used by the Hierarchy Manipulator (825) that is employed by 815. Event Visualization and Control also updates the set of positive and negative examples (820) and invokes the Similarity Query Engine (830) to aid in finding other positive and negative examples. Doing so requires specifying numerical distances between patterns, which 800 specifies through interactions with the Distance Calculator (835), a component that is invoked by the Similarity Query Engine.

Fig. 8, 9, and 10 illustrate how analysts interact with the system in our invention to construct correlation rules. Fig. 8 illustrates a user interface that can

be employed to achieve box 500 in Fig. 4. Seen here is a scatter plot. The x-axis is time and the y-axis is the host from which an event originated. The latter is categorical and so can be ordered in any manner that is convenient for visualizing patterns. The user has selected a set of events between times 5 and 9 and for hosts C, B, and A. This is an example of an underlying concept that the analyst knows but cannot articulate in a computer encoded manner. For the purposes of this example, the true concept contains all events that are on subnet 15.2.3 and for which the rate of the event set (count divided by time) exceeds .5/second.

Fig. 9 displays the results of steps 505, 510, and 520 in Fig. 4. Here, the machine has inferred from the example of Fig. 8 the concept that events originate from a Hub and that the rate exceeds .75/second. We see two new patterns that have been found by the machine that are consistent with this concept.

Fig. 10 shows how the analyst accomplishes step 530 in Fig. 4. The analyst has edited the patterns found by the machine to be consistent with the analyst's concept. Specifically, in one case an event is excluded; in the other, a previously excluded event is included.

CLAIMS

1. System for containing an event visualization, data access component, event group selection and editing, a similarity query, and a concept learner component to provides for learning situations.

2. Method for the system in claim (1) for learning situations that includes the steps of marking event groups as positive and negative examples, (2) inferring concepts from the examples, and (3) translating this into the left-hand side of a rule.

3. A method as in claim 2 in which the learning step includes: (a) learning an initial concept, (b) determining if this is statistically significant given the data, (c) querying historical event data for similar event groupings, and (d) allowing the user to critique the result.

4. Concept hierarchy used by the method in claim (2) that includes hierarchies for hosts and event types.

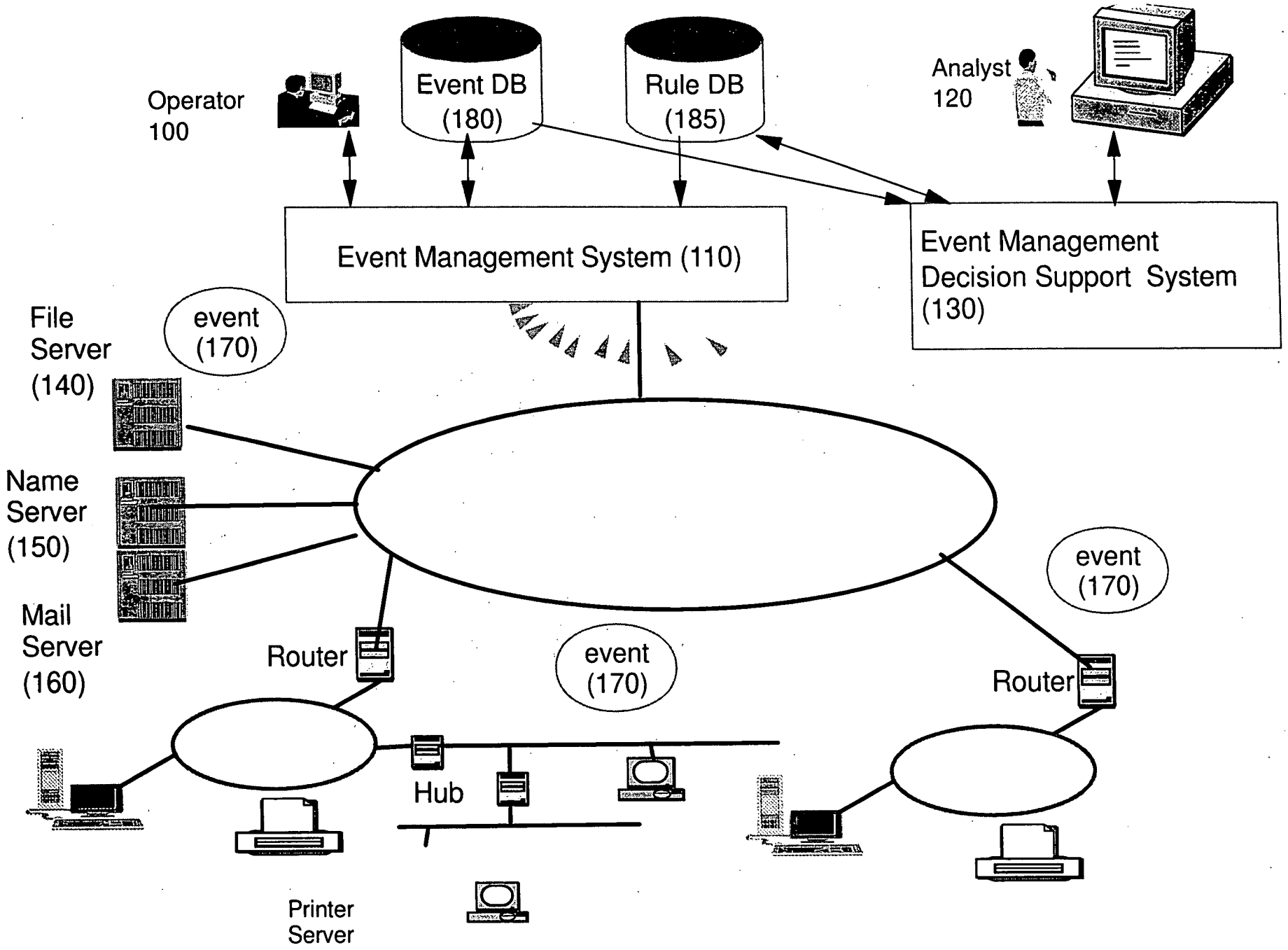

Operator
100
Event DB
(180)
Rule DB
(185)
Analyst
120
Event Management System (110)
Event Management
Decision Support System
(130)
File
Server
(140)
event
(170)
Name
Server
(150)
Mail
Server
(160)
Router
event
(170)
event
(170)
Router
Hub
Printer
Server

Fig. 1

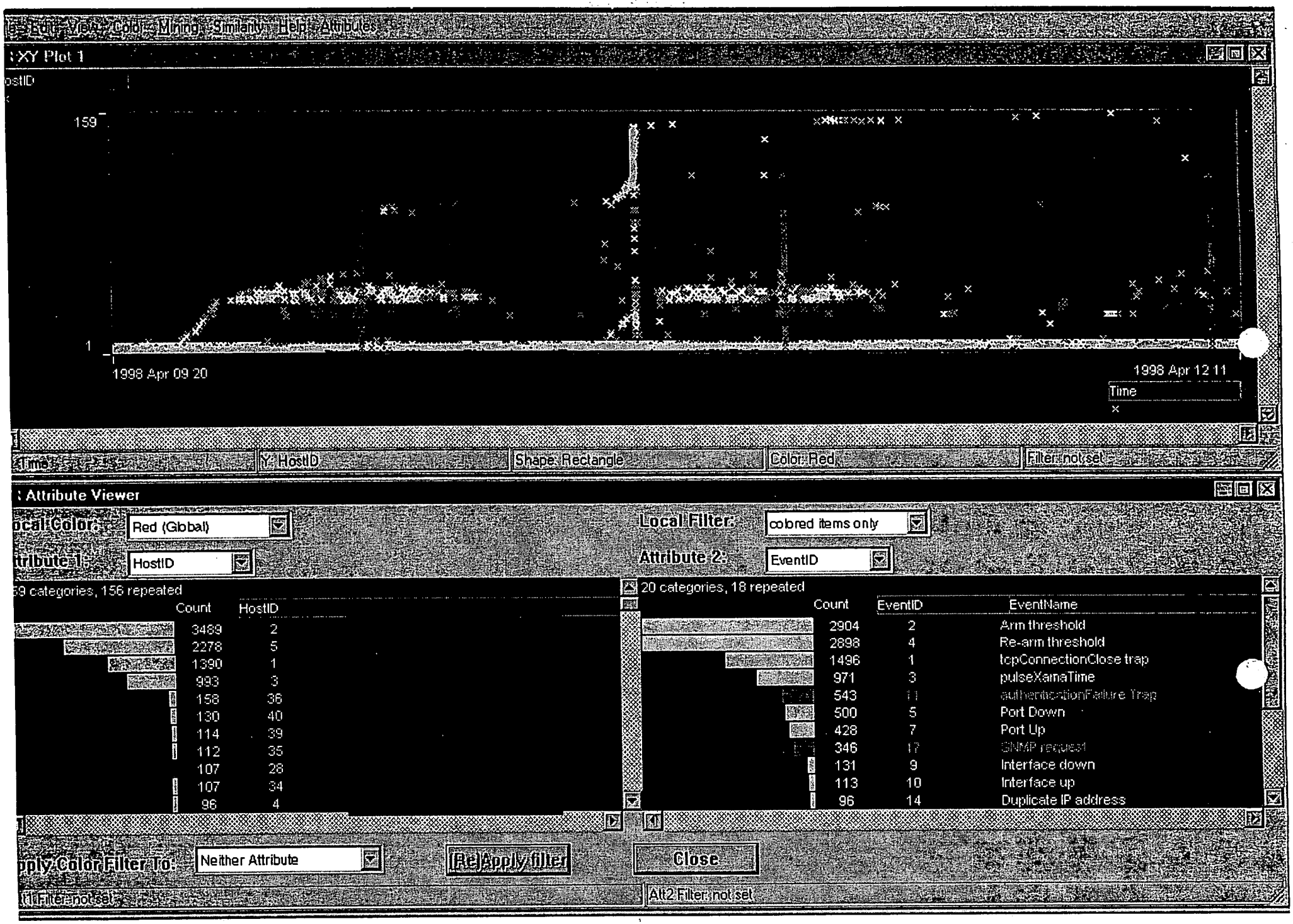
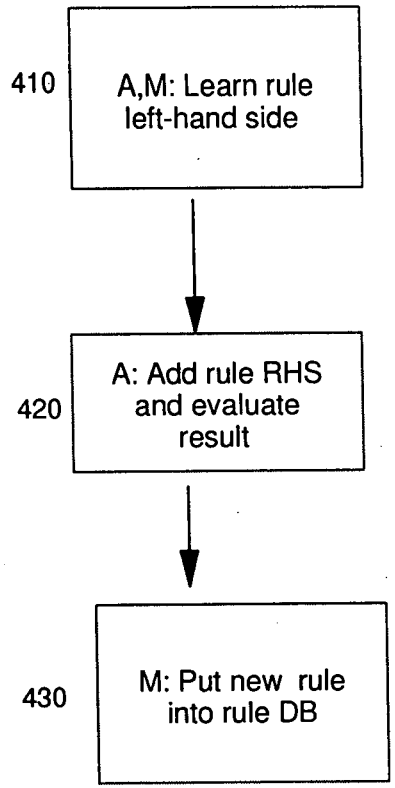

Edit View Color Mining Similarity Help Attributes
XY Plot 1
159
1
1998 Apr 09 20
1998 Apr 12 11
Time
Y: HostID
Shape: Rectangle
Color: Red
Filter: not set
Attribute Viewer
Local Color:
Red (Global)
Local Filter:
colored items only
Attribute 1:
HostID
Attribute 2:
EventID
59 categories, 156 repeated
Count
HostID
3489 2
2278 5
1390 1
993 3
158 36
130 40
114 39
112 35
107 28
107 34
96 4
20 categories, 18 repeated
Count
EventID
EventName
2904 2 Arm threshold
2898 4 Re-arm threshold
1496 1 tcpConnectionClose trap
971 3 pulseXamaTime
543 11 authenticationFailure Trap
500 5 Port Down
428 7 Port Up
346 17 SNMP request
131 9 Interface down
113 10 Interface up
96 14 Duplicate IP address
Apply Color Filter To:
Neither Attribute
(Re)Apply filter
Close
Alt1 Filter: not set
Alt2 Filter: not set

Fig. 2



A=Analyst
M=Machine

Fig. 3

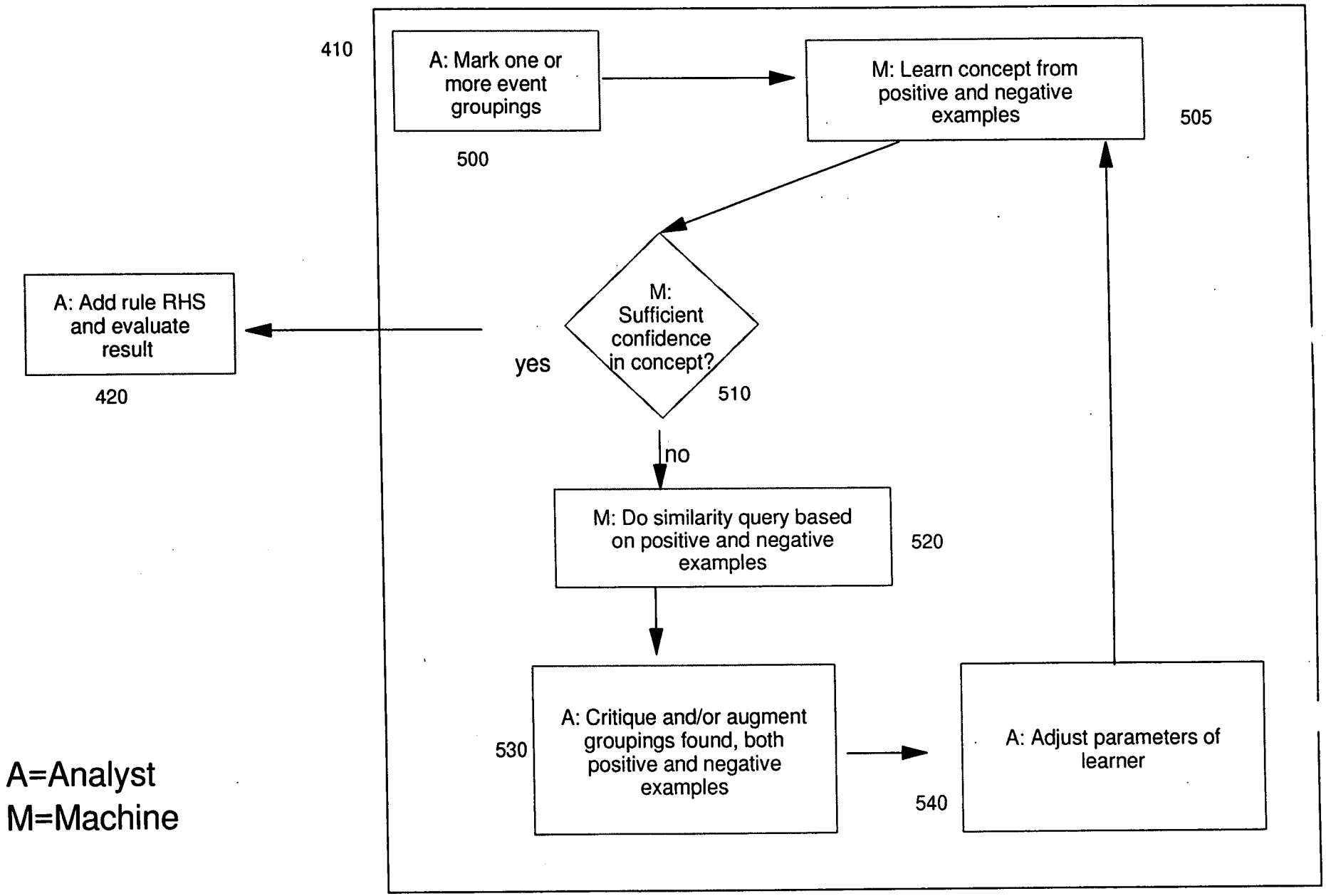

410
A: Mark one or more event groupings
500
M: Learn concept from positive and negative examples
505
M: Sufficient confidence in concept?
510
yes
A: Add rule RHS and evaluate result
420
no
M: Do similarity query based on positive and negative examples
520
A: Critique and/or augment groupings found, both positive and negative examples
530
A: Adjust parameters of learner
540
A=Analyst
M=Machine

Fig. 4

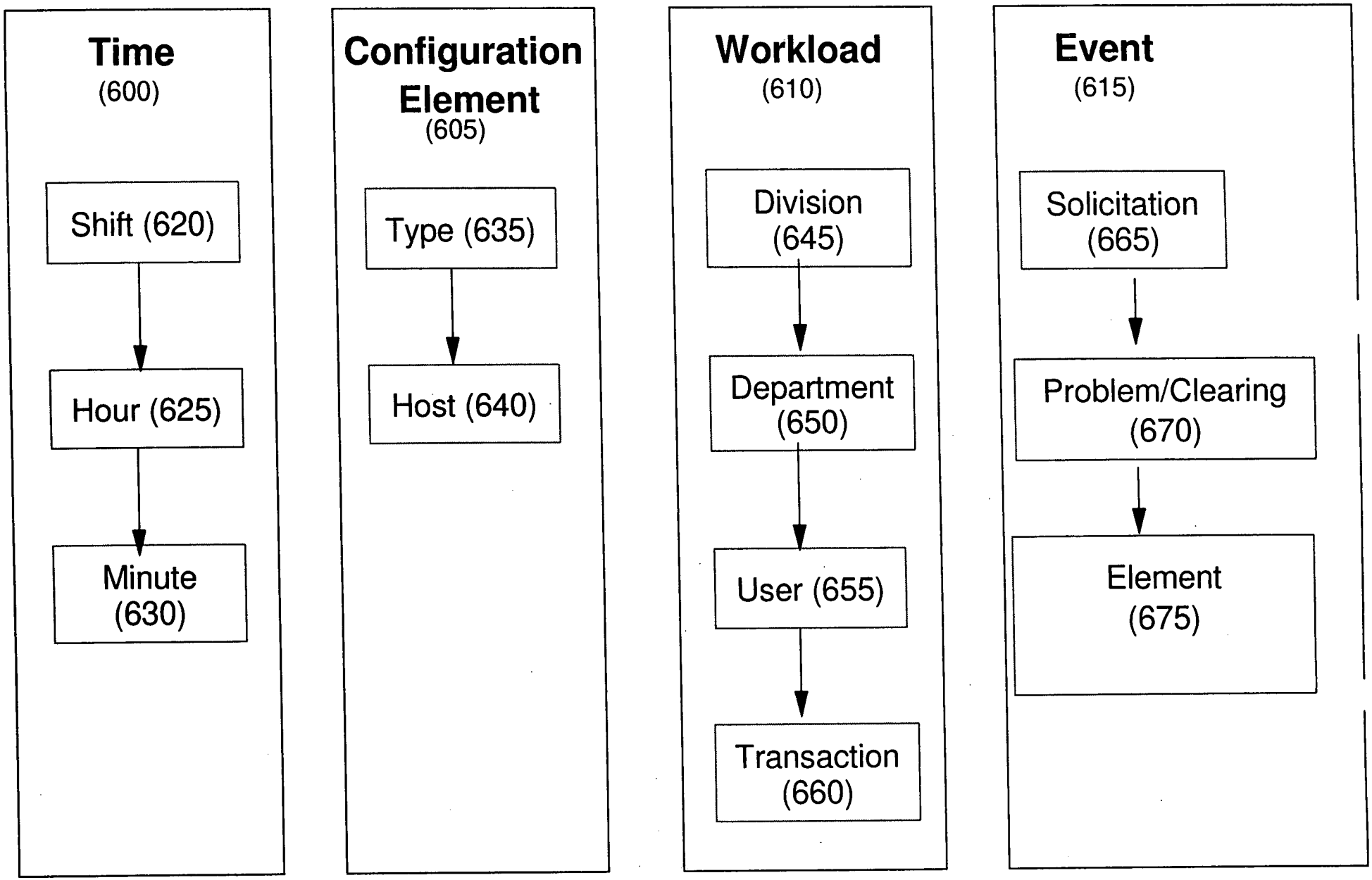

Time
(600)
Shift (620)
Hour (625)
Minute (630)
Configuration Element
(605)
Type (635)
Host (640)
Workload
(610)
Division (645)
Department (650)
User (655)
Transaction (660)
Event
(615)
Solicitation (665)
Problem/Clearing (670)
Element (675)

Fig. 5

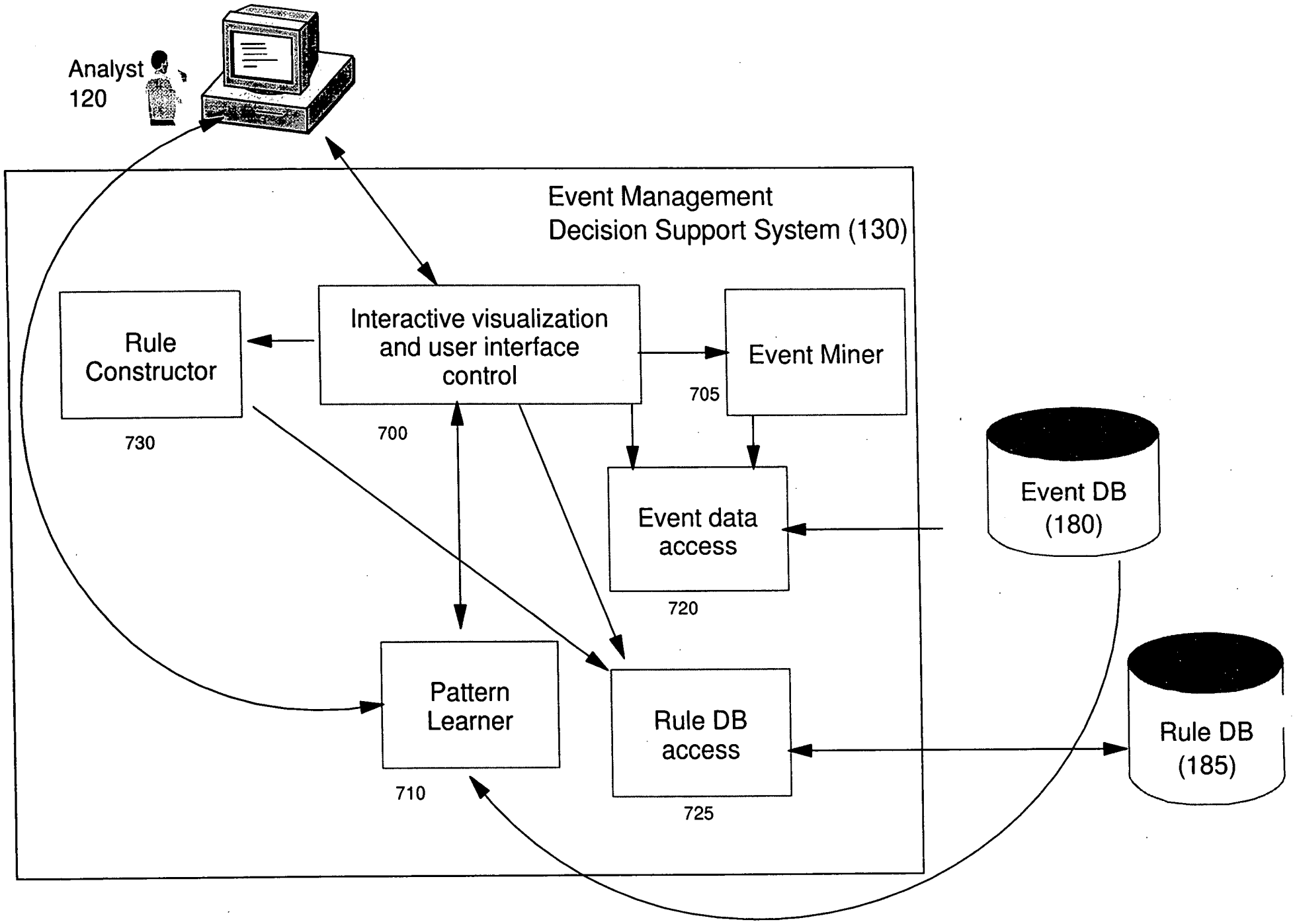

Analyst
120
Event Management
Decision Support System (130)
Rule
Constructor
730
Interactive visualization
and user interface
control
700
Event Miner
705
Event data
access
720
Event DB
(180)
Pattern
Learner
710
Rule DB
access
725
Rule DB
(185)

Fig. 6

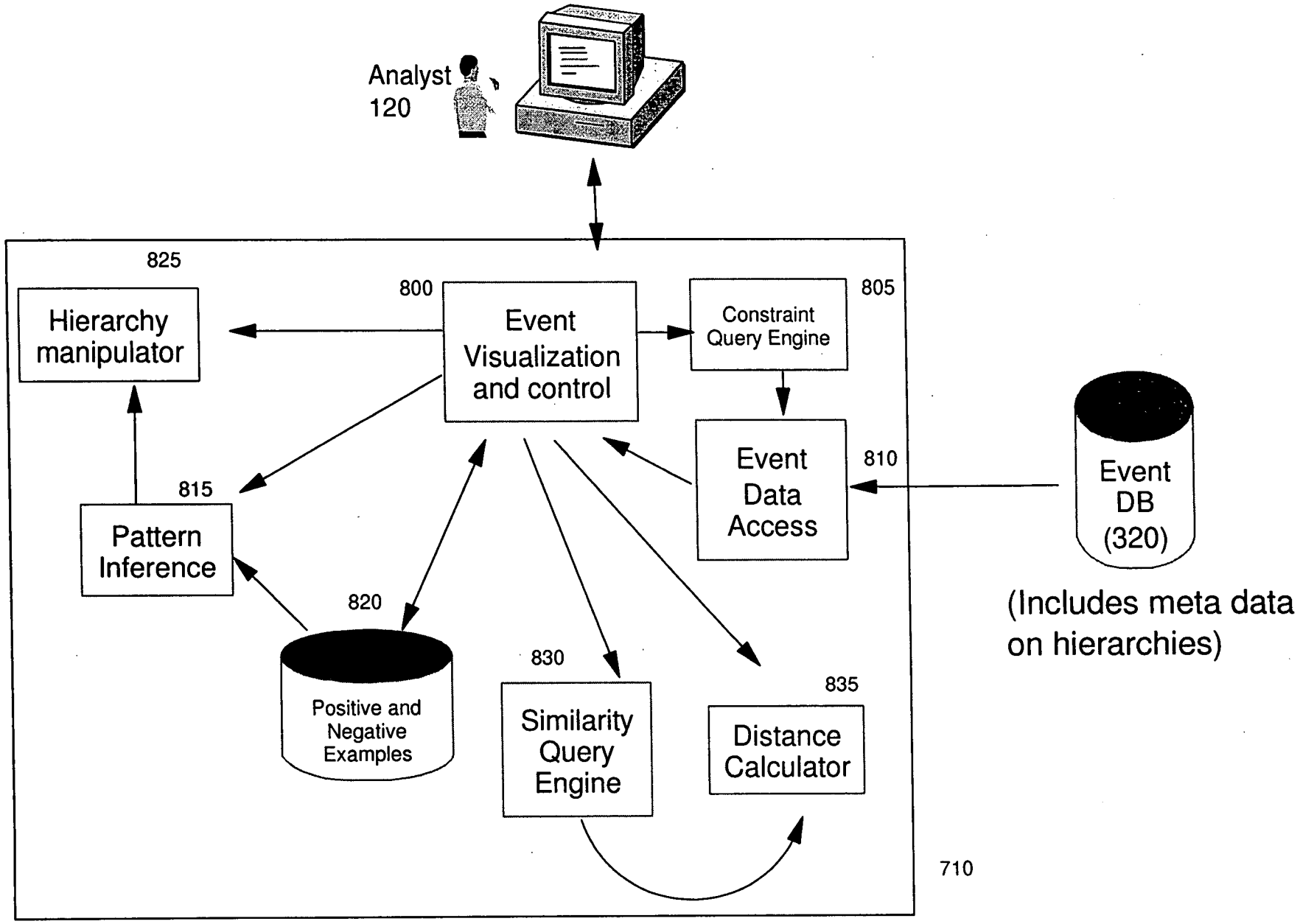

Analyst
120
825
800
Event
Visualization
and control
805
Constraint
Query Engine
Hierarchy
manipulator
Event
Data
Access
810
Event
DB
(320)
815
Pattern
Inference
820
Positive and
Negative
Examples
830
Similarity
Query
Engine
835
Distance
Calculator
710
(Includes meta data
on hierarchies)

Fig. 7

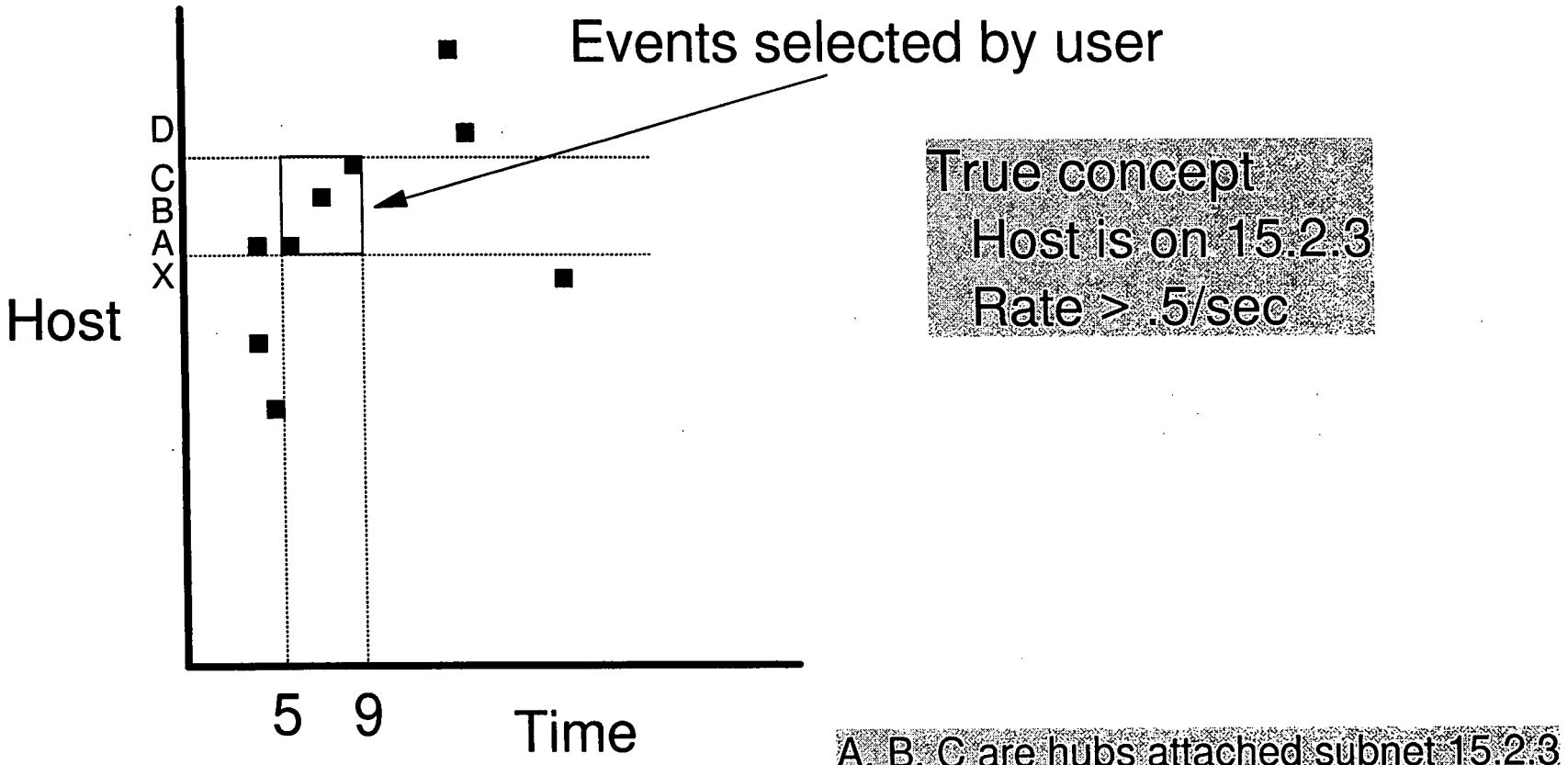

Events selected by user
D
C
B
A
X
Host
5
9
Time
True concept
Host is on 15.2.3
Rate > .5/sec
A, B, C are hubs attached subnet 15.2.3
X is not a hub and is on 15.2.3
D is a hub but is not attached to 15.2.3

Fig. 8

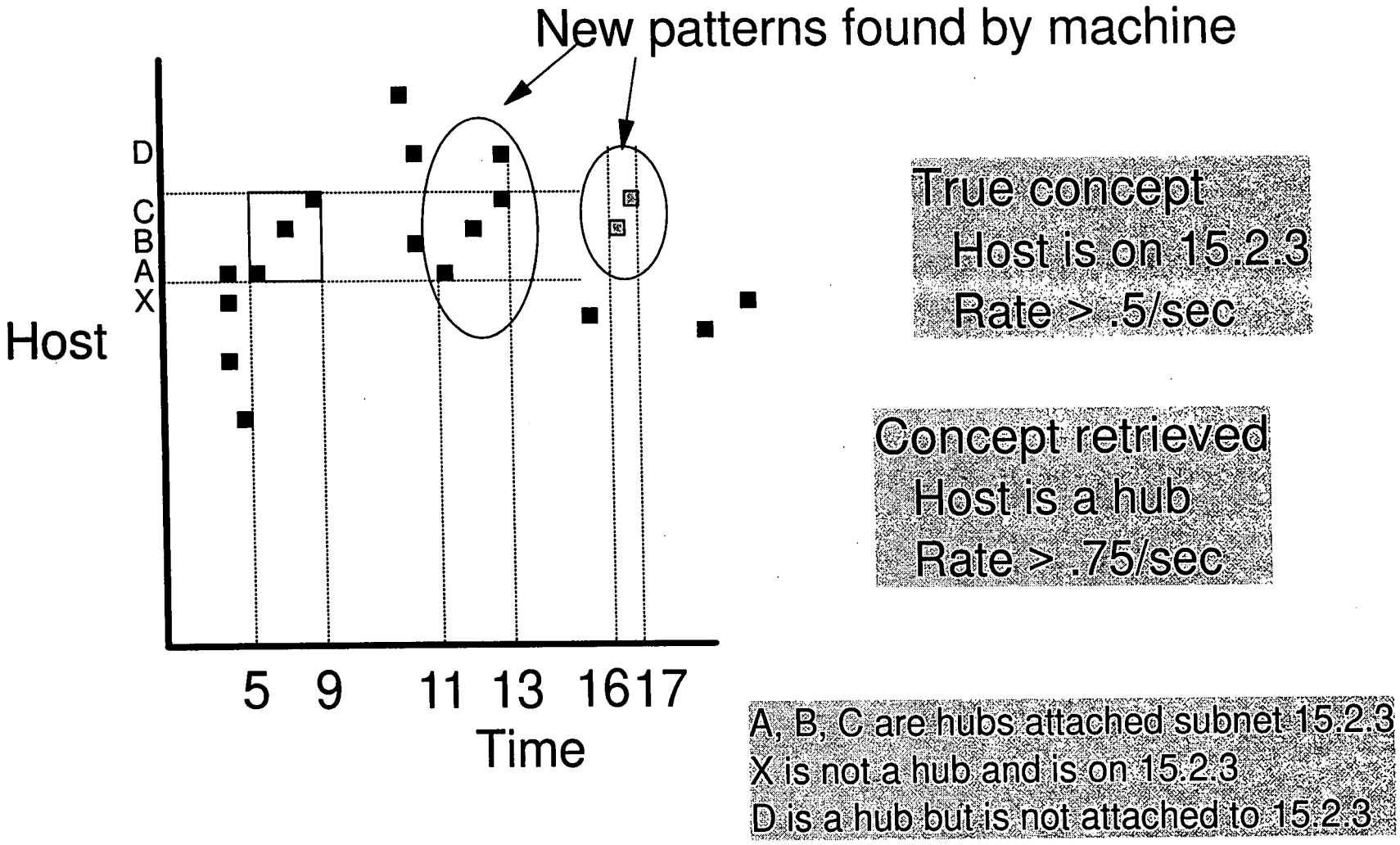

New patterns found by machine
Host
D
C
B
A
X
5
9
11
13
16
17
Time
True concept
Host is on 15.2.3
Rate > .5/sec
Concept retrieved
Host is a hub
Rate > .75/sec
A, B, C are hubs attached subnet 15.2.3
X is not a hub and is on 15.2.3
D is a hub but is not attached to 15.2.3

Fig. 9

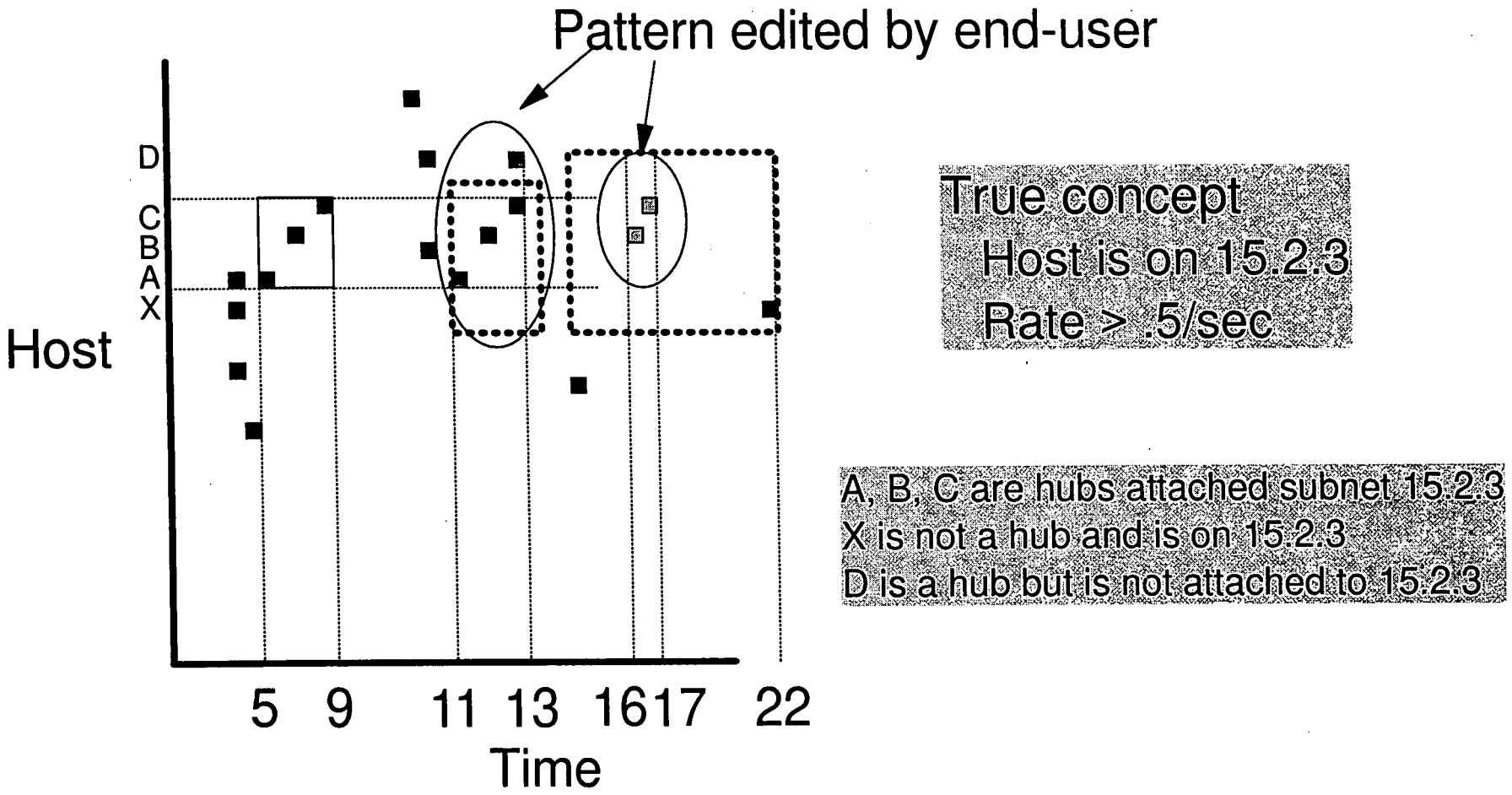

Pattern edited by end-user
Host
D
C
B
A
X
5
9
11
13
16
17
22
Time
True concept
Host is on 15.2.3
Rate > .5/sec
A, B, C are hubs attached subnet 15.2.3
X is not a hub and is on 15.2.3
D is a hub but is not attached to 15.2.3

Fig. 10